

A person is holding a smartphone, and the entire image is covered with a semi-transparent blue overlay. The text 'Oobee iOS SDK' and 'Interface Definition' is centered over the image.

# Oobee iOS SDK

## Interface Definition

**GEOCOMPLY.**

Version 1.0

01 May, 2019

*Document is for distribution only to intended or authorized recipients. © GeoComply 2017. All rights reserved.*

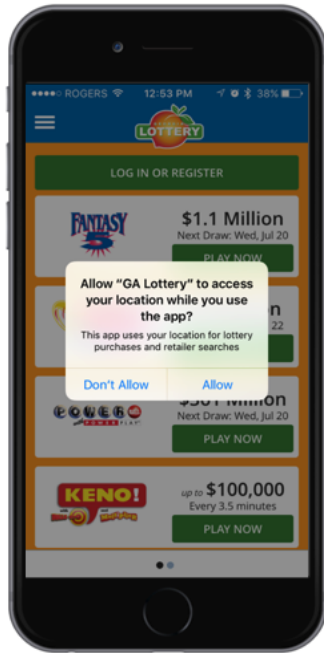
## DOCUMENT HISTORY

Date (yyyy-mm-day)	Vers.	Remarks
2019-05-01	1.0	■ Initial document

# TABLE OF CONTENTS

<b>1 OVERVIEW</b>	<b>4</b>
<b>2 REGULATORY CONSIDERATIONS &amp; REQUIREMENTS</b>	<b>4</b>
<b>3 INTEGRATION SCENARIOS</b>	<b>5</b>
3.1 WHEN SHOULD WE TRIGGER A GEOLOCATION?	6
3.2 GEOLOCATION REQUEST TIME AND TIMEOUT	7
3.2.1 User Experience during a geolocation	7
3.3 UX AND ERROR MESSAGES	8
3.3.1 Client Error Codes (6xx codes)	8
3.3.2 Rules Engine Errors	9
<b>4 INTEGRATION DETAILS</b>	<b>10</b>
4.1 NOTIFICATION SERVICE EXTENSION	11
4.2 EXAMPLE	11
<b>5 API REFERENCE</b>	<b>12</b>
5.1 OOBELIBRARY CLASS	12
5.1.1 Instance Methods	12
5.1.1.1 runOnPort:onComplete:	12
5.1.3.2 processToken:completionHandler:	12
5.1.3.3 processOobeeAPN:completionHandler	13
5.2 OobeeError enum	13

# 1 OVERVIEW



This document is an interface specification for integrating Oobee iOS SDK into our client end user apps. Prior to reviewing or implementing aspects of this document the developer and project manager must fully review the Project Initiation Document to ensure all regulatory and UX considerations have been made prior to integration.

In regulated gaming markets geolocation systems have a big impact on the user experience and the player funnel.

## DISCLAIMER

The information contained within this document is and shall remain the property of GeoComply. This document is supplied in strict confidence and must not be produced in whole or in part, used in tendering or for manufacturing purposes or given or communicated to any third party without the prior written consent of GeoComply.

# 2 REGULATORY CONSIDERATIONS & REQUIREMENTS

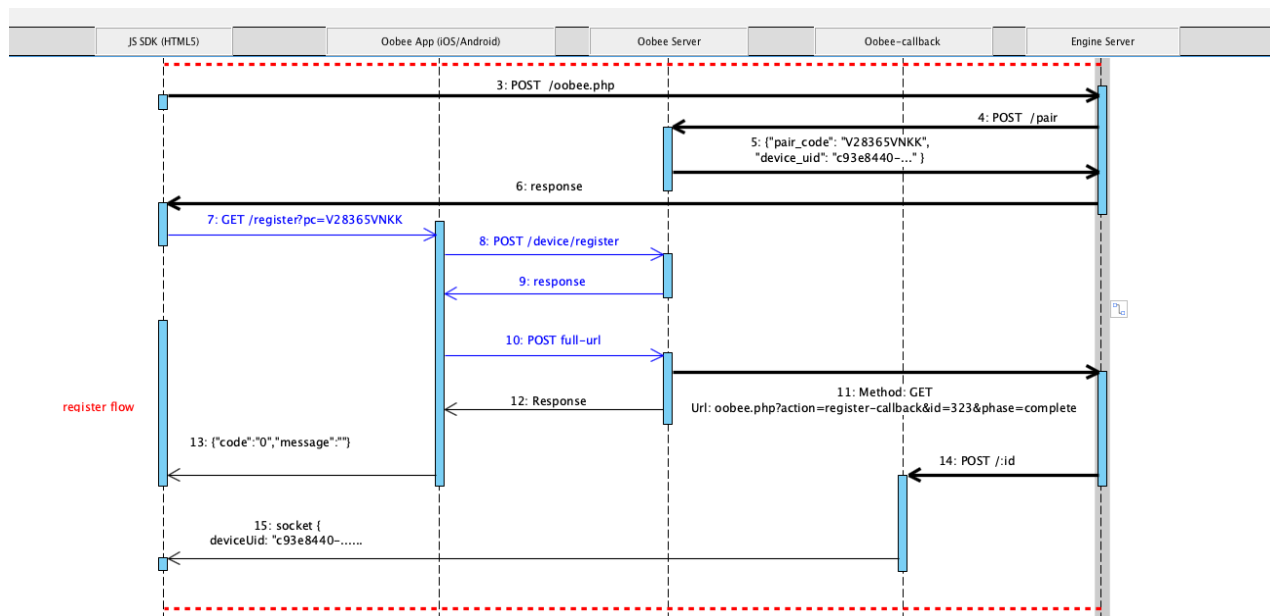
- GeoComply cannot guarantee to quality of old versions. All clients using the GeoComply systems in regulated markets should ensure they deploy latest releases on the release dates to maintain compliance and product integrity.
- All regulators have custom geolocation requirements. GeoComply's Customer Success team can advise on these specific requirements before integrating. Review of regulator requirements should have occurred during the Discover & Selection Phase mentioned in the Project Initiation Document. That said, this document will also describe considerations that must be thought about when integrating the solution for regulatory use cases.
- We highly recommend that the developer integrating GeoComply's systems discuss the market requirements with GeoComply prior to integrating.
- For operators deploying in highly regulated US markets you must request Administrative Access to ensure all required fraud detection works properly.
- Most regulated markets require GeoComply to locate a player prior to placing a bet or wager and periodically throughout the gaming session to ensure the player doesn't exit an allowed boundary during play.

- Regulators require GeoComply to provide report cases of suspicious activities. To ensure accuracy of these reports operator must provide us with the following attributes with the geo request:
  - User ID:** the unique identifier that identifies the user on the operator's system.
  - Reason:** Gaming Session (required) or other optional reasons such as: Deposit, Withdrawal, account opening, login.

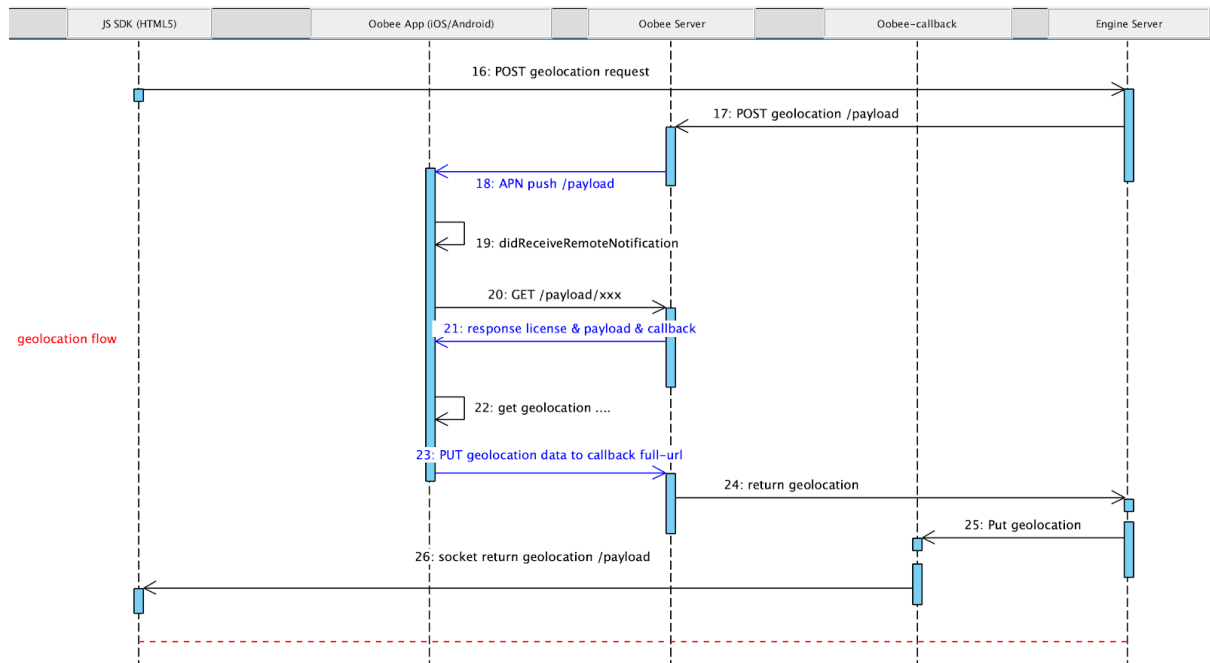
## 3 INTEGRATION SCENARIOS

The following diagram shows a register flow which return a deviceId for the next geolocation requests.

### Register Flow



## Geolocation Flow



<b>JS SDK</b>	The Oobee JS that handles Oobee functions on web browser.
<b>Oobee ios/android App</b>	The ios/android native apps that provides location and anti-fraud functions.
<b>Oobee Server</b>	The Oobee server
<b>Engine Server</b>	GeoGuard Engine Server
<b>Oobee callback</b>	The Oobee callback server

## Register Sequence

- Before triggering a geolocation request the web app needs to register it to Oobee Server. This process will return a device UUID to identify the device.
- The Oobee JS first makes a POST request to Engine Server to get a pair-code
- Then, Oobee JS uses that pair-code to register with native app via local server hosted by that native app.
- Oobee native app then registers that pair-code with Oobee Server.
- Finally, Oobee call back server will return a DeviceUUID to Oobee JS via a Socket.

## Geolocation Sequence

- To get geolocation, Oobee JS makes a POST request to Engine Server.
- A notification will be pushed to native app to wake it up and start geolocation.
- Finally, the geolocation result will be sent back to Oobee JS via socket.

### 3.1 WHEN SHOULD WE TRIGGER A GEOLOCATION?

There are two different types of geolocation transactions that should trigger the SDK:

- **Checkpoint of restricted areas:** geolocation should be triggered at the point where it needs to be decided to allow the users access to restricted services. For example, usually, the operator can trigger geolocation at the following places:
  - At login
  - Enter the game to play for real money
  - Placing a bet
  - Deposit or withdrawal
- **Periodic check:** besides checking the user's location when they enter a restricted area, we also need to do periodic checking of the user's position to ensure that the user has not moved out of the allowed zone during the play. In the GeoPacket response, the GeoComply Engine returns a value in the tag <geolocation\_in>. It is strongly recommended that the Operator App triggers another geolocation before the geolocate\_in period to ensure that the user is still in the allowed zone. You can also refer to the section "*Parsing Geopacket and Geolocation Token*" in the "*GeoComply Geolocation Response, XML Content*" for more details about the geolocate\_in and the geolocation token.

### 3.2 GEOLOCATION REQUEST TIME AND TIMEOUT

The geolocation time is divided into several components:

- **Location scanning time:** this scanning time is the time that the SDK waits for the location data to be available and collecting the necessary device information. The location scanning time is configurable via the Back Office.
- **Submitting data to the Engine:** is the time from the request being submitted to the GeoComply Engine server to the response being received back by the PLC and any retries if needed. Usually the processing time on the Engine itself is less than 1 second.

**Timeout:** We strongly advise that the Operator App should not have any timeouts that terminate the geolocation process of the SDK. The SDK guarantees that you will receive a callback defined by the SDK (e.g. OnGeolocationAvailable, OnGeolocationFailed) when the SDK finishes geolocation. So the Operator App should wait for the callback to be returned instead of terminating the geolocation process prematurely. If you still

want to set a timeout for the geolocation request, please consult our CS team for the best timeout value and method of integrating it.

*Please contact GeoComply if you want to adjust the geolocation time.*

### 3.2.1 User Experience during a geolocation

All of the geolocation requests in the SDKs are implemented to be run in the background. However, the Operator app can choose whether to perform the geolocation silently in the background or not. It depends on many factors:

- **For Geolocation Check For Accessing Restricted Areas:** depending on the regulations, usually, the Operator App should not allow the user to access the restricted areas until the geolocation is passed. Thus, the Operator app needs to show a progress indicator signifying that a geolocation is happening and blocking users from accessing the restricted area features until the geolocation is complete.
  - If the geolocation fails, an error message should be displayed to the user detailing the reason of failure. We recommend that you implement the messages that are supplied by GeoComply in order to provide the best possible UX. Please read more about this in sections 3.3. and 3.3.2.
- **For Periodic Check:** Operator App should run the geolocation in the background during the periodic check before the token has expired. If the app gets a failed result from the geolocation check, it should either retry or show an error message depending on the regulation requirements.

## 3.3 UX AND ERROR MESSAGES

When a geolocation results in an error or a failure, we will need to display the error message to the user. We need to differentiate between two different types of errors:

Errors from the SDK	Fail Reasons from Rules Engine
Returned by SDK in the OnGeolocationFailed() callback	Returned in the encrypted GeoPacket Response
Error Code is in the format of 6xx	There are no error codes but rule names that failed.
Error messages and whether to ask user for retry are returned by the SDK in the OnGeolocationFailed() callback.	From BO 4.3.0, the error message and retry will be inside the GeoPacket Response. For BO 4.2.0, please refer to the UX and Error Messaging for the details of each error message.



No transaction is made on the server

A failed transaction is made on the server.

### 3.3.1 Client Error Codes (6xx codes)

These are the error codes returned by the SDK when there is a problem completing a geolocation transaction. The error code will be returned by the SDK in the `didGeolocationFailed()` callback. In the callback, the error message and whether to display the retry button is also returned by the SDK. Developers should follow the recommendation from the SDK to display the error message accordingly.

Example of an error message for 6xx errors:



For the list of all error codes in the error messages, please refer to the “GeoComply Error Codes” documents.

Please refer to the [didGeolocationFailed](#) section in this document for more details.

### 3.3.2 Rules Engine Errors

When the Operator App requests a geolocation from the SDK, the SDK will first collect data from the device (e.g. wifi access points) and then submit these data to the GeoComply Engine Service. The Rules Engine will process the request based on a set of rules that are configured according to an agreed setup. If one or more of the rules failed, the details of the errors will be returned in the GeoPacket Response. There are 4 tags in the GeoPacket that you need to parse to get the details of the rules failure.

Please refer to the “GeoComply Geolocation Response, XML content” for the details explanation of each of the tags and the returned values from the rules:

- **error\_code**: if the transaction is a passed transaction, the `error_code` will be 0. If the transaction failed, the `error_code` is 1.
- **error\_message**: the error message tag contains the list of rules that failed in the transactions. The rule names in the list are predefined in the “GeoComply Geolocation Response, XML content” document. For example, the `error_message` can be “blocked\_software,boundary” which means that there are two rules that fail in the transaction: blocked software detection rule and boundary rule.

- **error\_summary**: rules failure are divided into 4 categories: `unconfirm_boundary`, `out_of_boundary`, `blocked_service`, `blocked_software`. So the `error_summary` tag can contain 1 or more tags from the 4 tags above mapping from the failed rules in the transaction.
- **error\_details**: the error details tag contains the details of the rules failure reasons. For example, for blocked software detection rule, it will contain the name, type, and display name of the blocked software that are detected. These additional data can help the Operator App to display specific error messages for different rules.



**IMPORTANT:** Since the rules settings can be changed in the GeoComply Back Office. It is highly recommended that operators should only rely on the `error_code` to pass or fail a user. Operators should NOT rely on the individual rules status to decide whether to fail users.

We also have a document “GeoComply Error Codes” that details all the error messages that the user should receive in different cases. It is highly recommended that the Operator should follow the recommended error messages closely to prevent user’s confusion.



**IMPORTANT:** In the BO 4.3.0, which will be released around March 2018, the GeoPacket will also contain the messages that should be displayed to the user. Thus the operator can just pass these messages directly from the GeoPacket instead of having to follow the specific document.

*More information will be added under “Troubleshooters” section in the “GeoComply Geolocation Response, XML content” once BackOffice 4.3.0 has been released.*

Below is an example of the error message:



## 4 INTEGRATION DETAILS

The distributed package will include:

1. GeoComplyOobee.framework. This is mandatory when using with Oobee supported websites.

2. GeoComplySDK.framework. This is mandatory when using Oobee supported websites as well as perform GeoComply Geolocation Flow.
3. OobeeLibraryResource.bundle. This is mandatory when using with GeoComplyOobee.framework.
4. SAMKeychain.framework. This is mandatory when using GeoComplyOobee.framework.

GeoComply Oobee iOS SDK is distributed as a framework that should be linked with the mobile app. The library name is **GeoComplyOobee.framework**.

To make available Oobee & GeoComply SDK classes and protocols to the mobile app, **GeoComplySDK.framework** should be also included in the application project.

Below is the list of main classes and protocols required for implementation of this scenario:

- **OobeeLibrary** class, for details see section 5.1.

Oobee iOS SDK uses standard iOS Apple frameworks internally. The mobile app must link to these frameworks for integration with GeoComply iOS SDK. Below is the list of frameworks:

- System Configuration Framework.
- Core Location Framework.
- Core Motion Framework.
- Core Bluetooth Framework.
- Security Framework.
- libresolv.tbd
- The app needs to add 3 usage descriptions into Info.plist file.
  - Location Privacy Usage Description:
    - If your app requires Always Location Usage, for iOS 11 NSLocationAlwaysAndWhenInUseUsageDescription and NSLocationWhenInUseUsageDescription keys must be included in Info.plist file. For iOS 10 and earlier, add NSLocationAlwaysUsageDescription key to your Info.plist file.
    - If your app requires When In Use Location Usage, add NSLocationWhenInUseUsageDescription key to the Info.plist file.
  - Motion Privacy Usage Description.
  - Bluetooth Peripheral Usage Description

Key	Type	Value
▼ Information Property List	Dictionary	(10 items)
Privacy - Bluetooth Peripheral Usage Description	String	Using bluetooth to detect Bluetooth adapter
Privacy - Location When In Use Usage Description	String	Use location to allow specific functions
Privacy - Motion Usage Description	String	Use motion activity to track indoor position.

Please refer to Apple Document for further information. [WWDC 17 - What's new in Location Technologies.](#)

Apple requires app developers to provide meaningful messages for location-related keys. Make sure to clearly describe the reason gathering location data.

#### Notes:

- From Xcode 7: The customer app can now enable **Bitcode** in Build Options of Build Settings by using Bitcode enabled version of the SDK inside the lib/BitcodeEnabled folder. If you do not want to use bitcode version, you need to use the version in the lib/BitcodeDisabled folder.
- Add **-ObjC** flag into "Other Linker Flags" in Project Build Settings as following

		General	Capabilities	Resource Tags	Info	Build Settings	Build Phases
Basic	Customized	All	Combined	Levels	+		
Other Librarian Flags							
Other Linker Flags		-ObjC					

### 4.1 NOTIFICATION SERVICE EXTENSION

GeoComply Oobee iOS works with Apple Push Notification (APN). Since Notification Service Extension (NSE) is tested with GeoComply Oobee app, the best practise is to create a new NSE for your app and add a few extra lines of handling code similar to handling APN in app. Please refer to our Demo folder for further details about creating an NSE and handling APN in NSE.

### 4.2 EXAMPLE

Please refer to *example* folder in the release package.

## 5 API REFERENCE

### 5.1 OOBEE LIBRARY CLASS

The class allows the operator app to subscribe to callback events and to trigger geolocation to receive geolocation data.

@interface OobeeLibrary : NSObject

#### 5.1.1 Instance Methods

##### 5.1.1.1 runOnPort:onComplete:

This method creates a local secure HTTP Server to support communication with Oobee supported websites.

```
- (void)runOnPort:(NSInteger)port
    onComplete:(void(^)(NSError* error))completion;
```

#### Parameters

*port*

The port for HTTP Server to be created on. See Remarks section for more details.

#### *completion*

Callback when finished. The error parameter will indicate if any error happens.

#### **Remarks**

Port number will be provided and managed by GeoComply to avoid conflict with other apps that also use GeoComplyOobee.framework.

### **5.1.3.2 processToken:completionHandler:**

This method triggers the handling process of Apple Remote Notification token.

- (void)processToken:(NSString\*)token

completionHandler:(OobeeUtilityHandler)handler;

#### **Parameters**

##### *token*

Device's remote notification token after removing "<", ">" and whitespaces.

##### *handler*

Callback when finished. OobeeUtilityHandler will include data and error parameters.

#### **Remarks**

From iOS 9, after reinstall the app, a new token will be created. In order for Oobee supported websites recognise new token, this method must be called after received a new token from Apple.

### **5.1.3.3 processOobeeAPN:completionHandler**

This method triggers handling process of Oobee's APN from Oobee Server.

- (void)processOobeeAPN:(NSDictionary\*)payload

completionHandler:(OobeeAPNHandler)completion;

#### **Parameters**

##### *payload*

The dictionary value from APN's key "geocomply\_oobee" which contains information to support geolocation from Oobee websites.

##### *completion*

Callback when finished. OobeeAPNHandler includes error and indicator flag parameters.

#### **Remarks**

The app should process APN and decide to call this method if "geocomply\_oobee" field appears.

## 5.2 OobeeError enum

**OobeeError** enumeration specifies error codes reported by GeoComply Oobee SDK. For handling instructions please refer to the Integration Overview document.

Enum Item Name	Value	Description
OobeeLibrary_NoError	0	No error.
OobeeLibrary_WrongRequestURL	128	Wrong local HTTP request format
OobeeLibrary_CannotConnectToGeoTechServer	129	The network connection is not available.
OobeeLibrary_NoResponseFromGeoTechServer	130	Timeout when request to GeoTech Server
OobeeLibrary_InvalidResponseFromGeoTechServer	131	Invalid response from GeoTech Server.
OobeeLibrary_UnknownError	300	Unknown error.
OobeeLibrary_InternalError	301	Internal library error.
OobeeLibrary_WrongAPNFormat	304	The APN payload format is invalid.
OobeeLibrary_DuplicatedAPN	305	Duplicated APN.
OobeeLibrary_MerchantIdNotFound	308	Merchant ID is not found in database.
OobeeLibrary_ProximityIsNotRequired	309	Proximity is not required on this device.
OobeeLibrary_ProximityTimeout	310	Proximity check timeout.
OobeeLibrary_InvalidReturnFromServer	311	Invalid response from server.
OobeeLibrary_CannotFindCallbackURL	312	Cannot find callback URL.
OobeeLibrary_SamePushToken	320	Same APN token.
OobeeLibrary_AlwaysNotGranted	321	Location Service Always permission is not granted
OobeeServer_DeviceWasAlreadyRegistered	12	Device is registered on server
OobeeServer_PairCodeInvalid	20	Invalid pair code.
OobeeServer_InvalidRequestData	91	Invalid request data to server.
OobeeServer_NoResponseData	100	No response data from server
OobeeServer_InvalidResponseData	101	Invalid response data from server.